

WEST Search History

[Hide Items](#)[Restore](#)[Clear](#)[Cancel](#)

DATE: Thursday, July 01, 2004

Hide?	<u>Set</u> <u>Name</u>	<u>Query</u>	<u>Hit</u> <u>Count</u>
		<i>DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ</i>	
<input type="checkbox"/>	L4	L3 and XML	2
<input type="checkbox"/>	L3	L2 and (parse or parsing or parser)	60
<input type="checkbox"/>	L2	20001223	554
<input type="checkbox"/>	L1	(database or (data adj2 bank) or (data adj2 base)) near8 (monitor or monitoring or detect or detecting) near8 (change or new or added)	987

END OF SEARCH HISTORY

[Previous Doc](#) [Next Doc](#) [Go to Doc#](#)
[First Hit](#) [Fwd Refs](#)



Generate Collection

L4: Entry 2 of 2

File: USPT

May 6, 2003

DOCUMENT-IDENTIFIER: US 6560611 B1

TITLE: Method, apparatus, and article of manufacture for a network monitoring system

Application Filing Date (1):
19990923

Detailed Description Text (10):

To store the data necessary for the RMS server and CAS server to perform their respective functions, a uniform standard data format may be employed, such as Standard Generalized Markup Language (SGML) or Extended Markup Language (XML). These languages provide efficient and flexible formats for storing data. In particular, unlike standard databases, the format can be modified quickly and easily to accommodate system updates and improvements. To process the data, any computer language may be used, such as Practical Extraction and Report Language (PERL). PERL is object-oriented, and provides a module for parsing and accessing SGML or XML data.

Detailed Description Text (16):

Secondary storage device 232 contains a database 234 that interfaces with informer engine 222. Database 232 contains a device file 236 that includes configuration information for RMS server 104 and specific information regarding each service to be monitored at client site 120. For example device file 236 may contain the IP address of the device 124, the IRQ of environmental sensor 244, or errors to locate when parsing log files associated with each service.

Detailed Description Text (29):

Each spawned checker software transmits a query, similar to the forker query, to informer engine 222 (step 512). The query is a request for additional details regarding the service to monitor on device 124 or sensor information. The query includes the address and port of the service checker software 226 will be monitoring. In the case of a security service, the query may include information regarding the log file to obtain from device 124. For example, checker software 226 may need to know a list of errors to look for in the log file. For example, checker software 226 may parse the log file for multiple well-known ICMP packets or invalid routes in a routing log file. Similar to step 508, informer engine 222 obtains this information from device file 234 and transmits the information to checker software 226. For example, if checker software 226 requests additional information on the web service in device 200.2.8.10, informer engine 222 may transmit a responses to checker software 226 that includes the directory to store tickets, the frequency at which to query each service, the duration of monitoring, and the contact person. Informer engine 222 may transmit the response as follows: Directory:/usr/ticket Frequency: 5 minutes Duration: infinite Contact person: Joe Smith

Detailed Description Text (37):

In a third technique, checker software 226 may obtain log file 332 that corresponds to service 322 on device 124. In doing so, checker software 226 may parse log file 332 and locate a potential problem. For example, checker software 226 may receive a log file from a router that indicates that a particular route is not functioning.

Although the router is responding, the log file would indicate an error.

Detailed Description Text (43):

As shown in FIG. 5C, once the ticket has been received by CAS 106, the dispatch process begins. The dispatch process begins, for example, by receiving ticket 600 at CAS 106 (step 536). Upon receipt of the ticket, receiver process 250 parses the ticket and uses the information in the ticket to query accounting engine 248 for information on where to place the pending ticket (step 538). For example, receiver software 250 may query accounting engine 248 with the IP address and port number of the service that is nonresponsive. Accounting engine 248 queries administrator file 262 for information regarding the service and responds with the location for ticket 600 in ticket file 260.

Detailed Description Text (51):

These reports may be used to help detect patterns in problems experienced by a device which may in turn lead to the detection of larger scale problems, such intrusion or security breeches or network traffic anomalies. For example, if a series of tickets indicate that a security log file on an NT server has a flood of ICMP packets, a report may be created to locate all of the tickets that indicate this problem. One skilled in the art will appreciate that reporter software 254 may access and parse ticket file 260 with well-known programs written in languages such as SQL.

CLAIMS:

7. The method of claim 6, wherein obtaining additional information further includes the steps of: requesting a log file associated with the service; receiving the log file; and parsing the log file for information regarding the status of the service.

13. The method of claim 12, further including the steps of: monitoring the database for a new data record; and spawning a process to dispatch the new data record to an administrator.

17. The method of claim 1, wherein notifying an accounting server further includes the step of: transmitting a notification to the accounting server, wherein the notification is in an SGML format, XML format or comma delimited text format.

[Previous Doc](#) [Next Doc](#) [Go to Doc#](#)

[Previous Doc](#) [Next Doc](#) [Go to Doc#](#)
[First Hit](#) [Fwd Refs](#)

☐ [Generate Collection](#)

L4: Entry 1 of 2

File: USPT

Mar 16, 2004

DOCUMENT-IDENTIFIER: US 6708189 B1
TITLE: Computer file transfer system

Abstract Text (1):

A method and apparatus are provided for converting a data file received by an automated publishing system from a source for use within one of a plurality of presentation spaces of the automated publishing system. The method includes the steps of parsing the data file to recover an identifier of the source and an information content to be used by the automated publication system and identifying a presentation space of the plurality of presentation spaces for the parsed file based upon the identifier of the source and reformatting the information content of the parsed file for use within the identified presentation space based upon a set of predetermined attributes associated with the identified presentation space.

Application Filing Date (1):
20000327

Brief Summary Text (11):

A method and apparatus are provided for converting a data file received by an automated publishing system from a source for use within one of a plurality of presentation spaces of the automated publishing system. The method includes the steps of parsing the data file to recover an identifier of the source and an information content to be used by the automated publishing system and identifying a presentation space of the plurality of presentation spaces for the parsed file based upon the identifier of the source and reformatting the information content of the parsed file for use within the identified presentation space based upon a set of predetermined attributes associated with the identified presentation space.

Detailed Description Text (31):

The controlling script serve two functions. Under a first function, a first set of the scripts function to control parsing of the hot files into its constituent parts. The presence of those constituent parts may then be used in conjunction with the business logic of the user to trigger reformatting decisions necessary to allow the reformatted file to conform to a presentation structure. Parsing may also be used to collect and format an identifier of the data source into meta data that may be used to identify a destination for the file. Identification of a destination for the file also inherently results in the identification of attributes, which the file must possess to be used within the destination space.

Detailed Description Text (41):

If there are more documents to be processed, the processor 46 determines the access needs of the document 108. Determining the access needs for the document may mean determining what format the document was originally created under and converting 110 the document back to a standard format (e.g., ASCII, XML, etc.).

Detailed Description Text (42):

In determining the access needs of the document, the processor 46 first decides what "type" file is involved (e.g., MS Word, ASCII file, QuarkXPress tag file, HTML document, XML document, ~~Word~~ Now file, WordPerfect file, etc.). Once the processor

46 determines what type of file is involved, the processor 46 applies the appropriate filter to read the file.

Detailed Description Text (43):

Using the appropriate filter, the processor 46 reads the contents of the file and begins parsing the file to extract specific items which the end user has requested that the processor 46 identify. Parsing the file may simply mean breaking the file up into character strings determined by a parsing criteria based upon a data file descriptor. Parsing may be performed by finding tabs, line returns, a specific number of characters, start of next character, defined text markers, invisible text markers, the end-of-the-file, etc.). The end user also has the option to retain source attributes about the extracted item if the user wishes to preserve this information.

Detailed Description Text (44):

The parsed data and extracted information may be directly used to reformat the parsed data based upon the business logic of the user. Alternatively, the parsed data may be stored as a parsed file under a standard, data neutral format (e.g., XML file, QuarkXPress file, etc.).

Detailed Description Text (45):

Where the hot file is to be stored as a parsed file under XML (e.g., DeskNetAPS-XML), the formatting which has been filtered from the source file is inserted into the XML file as mark-up language. The mark-up language of the XML file may be used as markers for identifying those features of the file (e.g., tabs, carriage returns, end of paragraph, end of section, etc.) which become important for reformatting.

Detailed Description Text (49):

The parsing operation may be used to recognize titles and an authors name based upon the provided format. The recognized information may be identified as reference named articles and encoded as meta data associated with the parsed file.

Detailed Description Text (50):

Operating under control of the software robot, the processor 46 stores the reference named article as meta data along with the parsed dataset. The processor 46 will continue to parse the file, perform action steps on each extracted item and save the processed extracted data into the dataset as either meta data or mark-up inserts until the file has been completely processed.

Detailed Description Text (69):

An asset management system (e.g., DeskNetAPS-MediaBank) is provided for archiving graphical media (e.g., under MS Word, XML, etc.). A user may enter descriptive information about the files as meta data. Search capabilities are provided for specific files based upon search terms matching information stored as meta data or content.

Detailed Description Text (86):

Middleware Monitoring Another Database for Data Change Example

Detailed Description Text (87):

The software robot monitors an Oracle database for any data changes that might effect the creation of a product. The software robot sees through its monitoring that some data has changed in the Oracle database and performs an SQL query to get the data (Raw material) that has changed. This data will be the layout name and due date. The software robot will parse the incoming data from the oracle database (db) into data sets.


Detailed Description Text (103):

In this example the software robot will make an EPS (image file) out of one of the

layouts and export it to a folder on a file server. The software robot will also open an article and manipulate and convert it to a format (e.g. HTML, XML, etc.) based upon the attributes of and the business logic associated with a destination presentation space (e.g. internet, intranet, world wide web, etc.). The converted content can then be immediately posted to the destination presentation space.

CLAIMS:

1. A method of converting a data file received by an automated publishing system from a source for use within one of a plurality of presentation spaces of the automated publishing system according to a business logic of an organization using the publication system, such method comprising the steps of: parsing the data file received from the source to recover an identifier of the source based upon the business logic of the organization using the publication system and an information content to be used by the automated publication system; identifying a presentation space of the plurality of presentation spaces for the parsed file based upon the identifier of the source and the business logic of the organization using the publication system; and reformatting the information content of the parsed file for use within the identified presentation space based upon a set of predetermined attributes associated with the identified presentation space.



22. A method of converting a data file received from a source for use at one of a plurality of destinations according to a business logic of an organization using the converted data file, such method comprising the steps of: parsing the data file received from the source to recover an identifier of the source based upon the business logic of the organization using the converted data file and an information content; identifying a destination of the plurality of destinations for the parsed file based upon the identifier of the source; identifying a set of predetermined attributes associated with the identified destination based upon the recovered identifier of the source; and reformatting the information content of the parsed file for use at the identified destination based upon the identified set of predetermined attributes.

31. The method of converting data files as in claim 25 wherein the step of formulating the set of formatting requirements further comprises preparing the file for storage as an HTML or XML construct.

35. A method of converting a data file received from a source for use at a destination according to a business logic of an organization using the converted data file, such method comprising the steps of: parsing the data file received from the source to recover an identifier of the source based upon the business logic of the organization using the converted data file, an identifier of a format of the data file and an information content; identifying a destination for the parsed file based upon an identifier of the source; retrieving a set of formatting requirements based upon the identified destination; and reformatting the information content of the parsed file for use at the destination based upon the identified format of the data file and the set of formatting requirements of the destination.

37. The method of converting a data file as in claim 35 wherein the step of parsing the file further comprises applying a formatting filter to recover formatting information.

38. The method of converting a data file as in claim 37 wherein the step of applying a formatting filter further comprises converting the filtered file into an XML file with the filtered formatting information inserted as mark-up language.

39. An apparatus for converting a data file received by an automated publishing system from a source for use within one of a plurality of presentation spaces of the automated publishing system according to a business logic of an organization using the publication system, such apparatus comprising: means for parsing the data

file received from the source to recover an identifier of the source based upon the business logic of the organization using the publication system and an information content to be used by the automated publication system; means for identifying a presentation space of the plurality of presentation spaces for the parsed file based upon the identifier of the source; and means for reformatting the information content of the parsed file for use within the identified presentation space based upon a set of predetermined attributes associated with the identified presentation space.

40. An apparatus for converting a data file received from a source for use at one of a plurality of destinations according to a business logic of an organization using the converted data file, such apparatus comprising: means for parsing the data file received from the source to recover an identifier of the source based upon the business logic of the organization using the converted data file and an information content; means for identifying a destination of the plurality of destinations for the parsed file based upon the identifier of the source; means for identifying a set of predetermined attributes associated with the identified destination based upon the recovered identifier of the source; and means for reformatting the information content of the parsed file for use at the identified destination based upon the identified set of predetermined attributes.

41. An apparatus for converting a data file received from a source for use at a destination according to a business logic of an organization using the converted data file, such apparatus comprising: means for parsing the data file received from the source to recover an identifier of the source based upon the business logic of the organization using the converted data file, an identifier of a format of the data file and an information content; means for identifying a destination for the parsed file based upon an identifier of the source; means for retrieving a set of formatting requirements based upon the identified destination for the identified source; and means for reformatting the information content of the parsed file for use at the destination based upon the identified format of the data file and the set of formatting requirements of the destination.

42. An apparatus for converting a data file received by an automated publishing system from a source for use within one of a plurality of presentation spaces of the automated publishing system according to a business logic of an organization using the publication system, such apparatus comprising: a parsing processor adapted to parse the data file received from the source to recover an identifier of the source based upon the business logic of the organization using the publication system and an information content to be used by the automated publication system; a destination processor adapted to identify a presentation space of the plurality of presentation spaces for the parsed file based upon the identifier of the source; and a formatting processor adapted to reformat the information content of the parsed file for use within the identified presentation space based upon a set of predetermined attributes associated with the identified presentation space.

[Previous Doc](#) [Next Doc](#) [Go to Doc#](#)

[Previous Doc](#) [Next Doc](#) [Go to Doc#](#)
[First Hit](#) [Fwd Refs](#)

**Generate Collection**

L2: Entry 2 of 9

File: USPT

Jun 15, 2004

DOCUMENT-IDENTIFIER: US 6751647 B1

TITLE: Method and apparatus for automated data exchange between a user computer and a provider computer using improved object-oriented programming components

Application Filing Date (1):
19980619

Brief Summary Text (12):

The present invention is directed generally to apparatus and methods for exchanging data between computers. In accordance with embodiments of the present invention, an improved component is provided for facilitating E-business processes over a network such as the Internet. Such embodiments promote a shift from the vendor-centric approach of existing E-business processes to a more flexible customer-centric approach.

Drawing Description Text (2):

FIG. 1 is a block diagram showing an E-business component for exchanging data between a user computer and a provider computer in accordance with an embodiment of the present invention.

Drawing Description Text (3):

FIG. 2 is a block diagram showing an E-business component for exchanging data between a user computer and a provider computer in accordance with another embodiment of the present invention.

Detailed Description Text (5):

Referring now to the drawings, FIG. 1 illustrates an apparatus for exchanging data between a user computer and a provider computer in accordance with a first embodiment of the present invention. The apparatus comprises a package 10 containing a component 12 and a documentation module 18. Package 10 can be implemented using any type of file capable of supporting components, such as a JAR (JAVA Archive) file, a CAB (Cabinet) file, or an FTP (File Transfer Protocol) file. Similarly, component 12 can be implemented using any component development technology, such as JAVAAPPLET, ACTIVEX and JAVABEAN. In this particular embodiment, component 12 includes a user application interface 14 for communicating with an application resident on the user computer (not shown), and a provider interface 16 for communicating with the provider computer (not shown). Component 12 includes a set of executable instructions configured to perform, for example, a business-related function. To distinguish it from existing component technology, package 10 can be referred to as an E-business component.

Detailed Description Text (15):

In this example, component 12 is a JAVABEAN component (see sample code in Appendix I), container application 20 is an EXCEL spreadsheet, documentation module 18 is a WORD97 file, and package 10 is a JAR file. Persons skilled in the art will recognize, however, that many variations are possible. Documentation module 18 conveys information to facilitate the customer's use of component 12. For example, documentation module 18 includes an "Introduction" section that advises the reader of the general function and capabilities of the E-business component. Documentation

module 18 also includes a section describing system requirements for running component 12, addressing platform requirements (e.g., operating system(s), processor, memory, disk space), software requirements (e.g., networking software), and security requirements (e.g., HTTP and TCP ports used for exchanging data with the provider computer). In addition, documentation module 18 includes a detailed discussion relating to the effective use of component 12, covering such areas as downloading and installation instructions, usage instructions, description of sample applications, intermittent connectivity support, instructions for in-process versus out-of-process operations, transactional support, and conditions that the manufacturer places on the use of its E-business component (e.g., agreement to limit use of the E-business component to interacting (whether directly or indirectly) with the manufacturer, agreement to display appropriate trademark and/or copyright notices). Appendix II contains sample text that may be included in documentation module 18.

Detailed Description Text (16):

By way of further illustration, FIG. 3 and FIG. 4 are block diagrams describing methods for using E-business components such as those described above to exchange data between two computers. FIG. 3 shows an embodiment of a method performed by a user of an E-business component, and FIG. 4 shows an embodiment of a method performed by a provider of an E-business component.

Detailed Description Text (17):

Referring now to FIG. 3, in accordance with this embodiment an E-business component is received by a user (Step 30). As discussed above with reference to other embodiments, the E-business component contains a component and a documentation module. In general, the component includes logic for exchanging data between a user computer and a provider computer, and the documentation module describes various rules, procedures, parameters, and so on for use of the component.

Detailed Description Text (19):

After the E-business component is received, the user configures an application on the user computer to exchange data with the provider computer using the E-business component (Step 35). Where the E-business component includes a container application, this configuration step can comprise little more than ensuring that the container application is supported by the user computer. On the other hand, the user may develop a customized application for using the E-business component. In such a case, this step may comprise configuring the customized application to interface with the component in accordance with the rules set forth in the documentation module. The properly-configured application can then be executed to exchange data between the user computer and the provider computer (Step 40).

Detailed Description Text (20):

FIG. 4 illustrates a method in accordance with another embodiment of the present invention. In accordance with this embodiment, an E-business component is configured by, for example, a provider of information or an intermediary (Step 50). As discussed above with reference to other embodiments, the E-business component contains a component and a documentation module. In general, the component includes logic for exchanging data between a user computer and a provider computer, and the documentation module describes various rules, procedures, parameters, and so on for use of the component. The E-business component is then provided to a user computer (Step 55). As discussed above, persons skilled in the art will recognize that distribution of the E-business component can be accomplished in a number of different ways. For example, the user can access a web site maintained by the provider of the E-business component and request that the E-business component be downloaded. Alternatively, the E-business component can be distributed on a portable storage medium, such as a diskette or a CD-ROM. The E-business component could also be pre-loaded on a computer prior to distribution of the computer to customers.

Detailed Description Text (21):

To further illustrate the type of expanded functionality facilitated by the present invention, FIG. 5 illustrates a system architecture in accordance with an embodiment of the present invention whereby an E-business component for exchanging data between two computers is configured to selectively operate in either an on-line or an off-line mode. In accordance with this embodiment, a first computer 60 and a second computer 70 are capable of communicating with one another over a network 80. In a particular implementation, network 80 comprises the Internet, but the present embodiment is equally applicable to other network configurations.

Detailed Description Text (22):

As shown in FIG. 5, first computer 60 has loaded therein an E-business component 62 such as those described above with reference to FIG. 1 and FIG. 2. First computer 60 also includes a local component 64 that governs access by E-business component 62 to a local database 66. Second computer 70 has loaded therein a remote component 72 that is configured to manage exchanges of data with first computer 60. Remote component 72 is in turn coupled to SAP 74 and remote database 76.

Detailed Description Text (23):

When a network connection exists between first computer 60 and second computer 70, E-business component 62 may receive information from, and provide information to, second computer 70 through remote component 72. Such exchanges of data may occur in either foreground or background mode, meaning a user of first computer 60 does not necessarily have to initiate the exchange or even be aware that it is occurring. Information received from second computer 70 is passed from E-business component 62 to local component 64, which in turn uses the information to update local database 66. In this way, information from second computer 70 can be stored in local database 66 for retrieval by E-business component 62 in response to a user inquiry even when the two computers 60, 70 are not connected. Likewise, when no network connection is established, E-business component 62 can store information in local database 66 for later transmission. To facilitate an arrangement of the type shown in FIG. 5, E-business component 62 can provide documentation identifying a local resource manager to be used for off-line storage (e.g., a database, a message queue, a text file), the component models that a container application should support (e.g., ACTIVEX, JAVABEANS), and the network communication protocols required (e.g., TCP/IP, DCOM, IIOP, RMI).

[Previous Doc](#) [Next Doc](#) [Go to Doc#](#)

[Previous Doc](#)[Next Doc](#)
[First Hit](#)[Go to Doc#](#)

Generate Collection

L4: Entry 2 of 12

File: PGPB

Aug 2, 2001

DOCUMENT-IDENTIFIER: US 20010011220 A1

TITLE: COMPUTER-BASED METHOD AND SYSTEM FOR AIDING TRANSACTIONS

Application Filing Date:19980219Detail Description Paragraph:

[0030] Though not represented in FIG. 1, each local representative has connectivity with other local representatives, banks, CA's, insurance companies, underwriters, etc. The local representative may need to provide for an electronic payment system as well as documentation, timestamping and other services.

Detail Description Paragraph:

[0037] An embodiment of a system for implementing the invention is shown in FIG. 4 and is composed of a supporting infrastructure made up of a plurality of local representatives. Each local representative may be some type of financial institution, such as a bank, a certification authority, an insurance company, an underwriter, etc. The supporting infrastructure may be composed of some or all institutions in a local area, in an entire country, a multinational region, or the entire world. The local representatives are linked together by appropriate communications channels having whatever degree of security is considered necessary.

[Previous Doc](#)[Next Doc](#)[Go to Doc#](#)

[Previous Doc](#) [Next Doc](#) [Go to Doc#](#)
[First Hit](#) [Fwd Refs](#)

[Generate Collection](#)

L4: Entry 4 of 12

File: USPT

Mar 5, 2002

DOCUMENT-IDENTIFIER: US 6353812 B1

TITLE: Computer-based method and system for aiding transactions

Application Filing Date (1):
19980219

Detailed Description Text (5):

Though not represented in FIG. 1, each local representative has connectivity with other local representatives, banks, CA's, insurance companies, underwriters, etc. The local representative may need to provide for an electronic payment system as well as documentation, timestamping and other services.

Detailed Description Text (12):

An embodiment of a system for implementing the invention is shown in FIG. 4 and is composed of a supporting infrastructure made up of a plurality of local representatives. Each local representative may be some type of financial institution, such as a bank, a certification authority, an insurance company, an underwriter, etc. The supporting infrastructure may be composed of some or all institutions in a local area, in an entire country, a multinational region, or the entire world. The local representatives are linked together by appropriate communications channels having whatever degree of security is considered necessary.

[Previous Doc](#) [Next Doc](#) [Go to Doc#](#)

[Previous Doc](#) [Next Doc](#) [Go to Doc#](#)
[First Hit](#) [Fwd Refs](#)

☐ [Generate Collection](#)

L4: Entry 6 of 12

File: USPT

Feb 16, 1999

DOCUMENT-IDENTIFIER: US 5873066 A

TITLE: System for electronically managing and documenting the underwriting of an excess casualty insurance policy

Abstract Text (1):

A computer-implemented system for managing the underwriting, quoting and binding by an insurance company of an excess casualty insurance policy for an insured having a primary insurance policy with a primary insurance limit amount. The system selects and stores at least one standard industrial classification code ("SIC code") associated with the insured and a primary carrier name or multiple carriers associated with the primary insurance policy. A plurality of SIC code records corresponding to a plurality of SIC codes are stored in a database. Each of the SIC code records are linked to underwriting guidelines established and filed by the insurance carrier. These criteria include guidelines related to minimum premiums, hazard rating, underwriting authority, and referral criteria. Primary insurance carrier public bureau rating records are also stored in the database. Each of the primary insurance carrier public bureau rating records includes a field for storing a rating code representing a financial stability rating associated with a primary insurance carrier. The system displays for the insurance carrier underwriter a plurality of candidate risk modifiers associated with the retrieved SIC code record, and for documenting and storing a selected risk modifier code and related underwriting criteria associated with the policy. The system develops the quotation using a detailed description of the insured's operation, the minimum premium amount information, the selected hazard code, the selected risk modifier code, primary insurance limits, and one or more attachment points.

Application Filing Date (1):

19970210

Brief Summary Text (11):

The present invention is directed to a computer-implemented system for managing the underwriting, quoting and binding by an insurance company of an excess casualty insurance policy for an insured having a primary insurance policy with a primary insurance limit amount. Means are provided for selecting and storing at least one standard industrial classification code ("SIC code") associated with the insured and a primary carrier name or multiple carriers associated with the primary insurance policy. A plurality of SIC code records corresponding to a plurality of SIC codes are stored in a database. Each of the SIC code records are linked to underwriting guidelines established and filed by the insurance carrier. These criteria include guidelines related to minimum premiums, hazard rating, underwriting authority, and referral criteria. Primary insurance carrier public bureau rating records are also stored in the database. Each of the primary insurance carrier public bureau rating records includes a field for storing a rating code representing a financial stability rating associated with a primary insurance carrier. Means are provided for retrieving a primary insurance carrier rating record associated with the insured from the database and for comparing the rating code stored in the primary insurance carrier rating record to an acceptable predetermined rating level. If the rating code associated with the primary insurance carrier is below the predetermined rating level, means are provided for

declining the quotation and/or initiating a home office referral process. Means are also provided for retrieving an SIC code record associated with the SIC code of the insured, for displaying for the insurance carrier underwriter hazard rating information linked to the SIC code record, and for receiving and storing one of a plurality of hazard codes selected by the insurance carrier underwriter in response to the displayed hazard rating information. Means are further provided for displaying for the insurance carrier underwriter a plurality of candidate risk modifiers associated with the retrieved SIC code record, and for documenting and storing a selected risk modifier code and related underwriting criteria associated with the policy. Finally, means are provided for developing the quotation using a detailed description of the insured's operation, the minimum premium amount information, the selected hazard code, the selected risk modifier code, primary insurance limits, and one or more attachment points.

[Previous Doc](#) [Next Doc](#) [Go to Doc#](#)

[Previous Doc](#) [Next Doc](#) [Go to Doc#](#)
[First Hit](#) [Fwd Refs](#)



Generate Collection

L4: Entry 7 of 12

File: USPT

Dec 29, 1998

DOCUMENT-IDENTIFIER: US 5855005 A

**** See image for Certificate of Correction ****

TITLE: System for electronically auditing exposures used for determining insurance premiums

Application Filing Date (1):
19960624

Brief Summary Text (4):

In order to determine the appropriate premium to charge an insured, an insurance company or underwriter setting the premium must have an accurate assessment of the insured's total exposure. An insured's total exposure is often based on criteria such as, for example, the total dollar volume of the insured's sales, the type of business engaged in by the insured, the total payroll of the insured, the number and types of vehicles used by the insured, etc. This exposure information may be provided by the insured to an insurance company or underwriter at the time that an insurance policy is initially quoted or, alternatively, through a preliminary audit conducted by the insurance company. At the time that a policy is up for renewal, it is desirable to have the ability to audit the insured (perhaps a second time if a preliminary audit was done) in order to confirm whether the insured's total exposure has changed and, if such a change has occurred, to adjust the insured's premium accordingly. In addition, at the time that a policy is canceled or expires, it is desirable to have the ability to audit the insured to confirm whether the insured's total exposure changed during the policy period and, if such a change occurred, to assess (or refund) any back premiums due as a result of the change in the insured's exposure.

[Previous Doc](#) [Next Doc](#) [Go to Doc#](#)

[Previous Doc](#) [Next Doc](#) [Go to Doc#](#)
[First Hit](#) [Fwd Refs](#)

☐ [Generate Collection](#)

L4: Entry 8 of 12

File: USPT

Dec 1, 1998

DOCUMENT-IDENTIFIER: US 5845256 A

TITLE: Interactive self-service vending system

Application Filing Date (1):
19971117

Brief Summary Text (3):

This invention relates to interactive self-service delivery systems and, more particularly, to insurance policy contract vending systems which operate automatically to interact with customers without the active intervention of insurance agents and underwriters, which normally operate off-line to limit line and central office computer time, and which provide a copy of an insurance policy contract signed by the customer to the insurance company.

[Previous Doc](#) [Next Doc](#) [Go to Doc#](#)

[Search Forms](#)[Search Results](#)[Help](#)[User Searches](#)[Preferences](#) 10 of 12[Logout](#)[Previous Doc](#)[Next Doc](#)[Go to Doc#](#)[First Hit](#)[Fwd Refs](#)

Generate Collection

File: USPT

May 16, 1989

DOCUMENT-IDENTIFIER: US 4831526 A

TITLE: Computerized insurance premium quote request and policy issuance system

Application Filing Date (1):

19860422

Detailed Description Text (21):

(6) A loss information directory, which notifies the appropriate underwriter of the occurrence of a loss, utilizes a loss data base within the central data base. There is also provided a directory for the claims department of a typical insurance company to notify the appropriate underwriter of an unusual loss when it occurs.

Detailed Description Text (24):

(9) A cancellation capability wherein an underwriter may execute cancellation transactions which, in the prior art were performed by a separate branch of the insurance company on instructions of the underwriter. In the present system, all policy requests which are not approved by the underwriter are either declined or canceled depending upon whether or not those policies have been issued. If the request requires pre-underwriting or has not yet been issued, the underwriter may decline all or part of the risk. If so, the declination and its reasons are entered in the declined file. If, on the other hand the policy was issued, the underwriter may decline a risk by requesting cancellation. This information also is entered in the declined file. In addition, the prospect file is updated to reflect any risk that has been declined.

[Previous Doc](#)[Next Doc](#)[Go to Doc#](#)

[Previous Doc](#) [Next Doc](#) [Go to Doc#](#)
[First Hit](#) [Fwd Refs](#)



Generate Collection

L3: Entry 20 of 60

File: USPT

Nov 26, 2002

DOCUMENT-IDENTIFIER: US 6487469 B1

TITLE: System and method for integrating schedule and design environments

Application Filing Date (1):

19991110

Brief Summary Text (8):

In another embodiment of the present invention, a method of project management integration includes detecting a change to data of a first database and obtaining first hierarchical data associated with the first database in response to the detected change. The method also includes obtaining second hierarchical data associated with a second database in response to the detected change, comparing the first hierarchical data to the second hierarchical data and identifying at least one difference between the first and second of hierarchical data, and automatically generating and sending an update command to the second database automatically in response to the at least one identified difference.

Brief Summary Text (9):

In yet another embodiment of the present invention, a system for automatically updating data of a first database in response to changes made to a second database includes a first interface in communication with the first database and a second interface in communication with the second database, the second interface operable to detect changes to data in the second database. The system also includes a comparison module in communication with the first and second interface, the comparison module operable to compare first hierarchical data associated with the first database to second hierarchical data associated with the second database in response to a change detected by the second interface. The comparison module is also operable to send update data to the first interface in response to comparing the first and second hierarchical data, the first interface being operable to change the data in the first database in response to the update data.

Detailed Description Text (13):

Integration module 50 includes software to monitor for and translate changes made to data in design database 20 or schedule database 30 and to determine whether or not such changes need to be reflected in any or both of the remaining databases. If a change does need to be reflected in one of the other databases, integration module 50 translates the hierarchical change in one database into an instruction that can be interpreted by the target database wherein such change is to be reflected. The instruction generated by integration module 50 in turn modifies the target database to reflect the change. Alternatively, the instruction generated by integration module 50 may merely provide an indication to a project manager, project member, or other user later accessing the target database that a change in that particular database is required. Such a request would indicate the change needed and any additional data relative to the change or entry thereof into the target database. As such, the hardware and software making up integration module 50 must provide monitoring, translation and instruction generating capabilities.

Detailed Description Text (17):

In general, interfaces 60, 70 and 80 monitor for changes to data in a respective

database, and pass along hierarchical data associated with a changed database to one of modules 90 and 100 in response to a change. Module 90 or Module 100 then compares that hierarchical data to hierarchical data associated with one of the remaining unchanged databases, determines what modifications need to be made to data in the unchanged database in order to synchronize the two sets of hierarchical data, and initiates such modifications. The modification or database synchronization may be achieved by generating an instruction related to the change made and sending this instruction to one or both remaining databases.

Detailed Description Text (19):

Interfaces 60, 70 and 80 are elements of integration module 50 that include any combination of hardware and software capable of monitoring changes to design database 20, schedule database 30, and ticket database 40 respectively, and further capable of communicating with such databases in order to obtain hierarchical data from such databases and send instructions to such databases. As such, interfaces 60, 70 and 80 communicate with databases 20, 30 and 40 across a data bus, public or private network, or other link or connection. As discussed, interfaces 60, 70 and 80 monitor databases 20, 30 and 40 respectively. Additionally, interfaces 60, 70 and 80 parse, receive, filter and translate data from databases 20, 30 and 40 for communication to first and second comparison modules 90 and 100. Interfaces 60, 70 and 80 also generate instructions to databases 20, 30 and 40.

Detailed Description Text (23):

Design interface 60 includes a monitor 110 in communication with design database 20 and a parser 120. A filter 130 is also in communication with parser 120 and additionally with a translator 140. Translator 140 is in communication with first comparison module 90 and a command generator 150. Command generator 150 is also in communication with design database 20. Elements of design interface 60 such as monitor 110, parser 120, filter 130, translator 140, and command generator 150 may be routines or software processes performed by one or more processing devices.

Detailed Description Text (24):

Monitor 110 monitors for and detects changes to design data of design database 20. Monitor 110 may monitor for such changes using a particular polling routine or technique or may monitor for such changes by waiting for interrupts generated by design database 20 to design interface 60. When monitor 110 identifies that a change has been made to design database 20, monitor 110 communicates the existence of such change to parser 120.

Detailed Description Text (25):

Parser 120 interrogates or otherwise reviews the design data of design database 20 to generate hierarchical data representative of the arrangement of design cells within design database 20. Parser generates hierarchical data by noting the relative position and structure of each design cell in design database 20. For example, a particular design cell expandable from a module within the design cell for a larger system, such particular design cell expanding to show three smaller modules, would have hierarchical data noting the identity of the particular design cell, the relationship of the particular design cell to that of the larger system, and the relationship of the particular design cell to each of the three smaller modules. In such a way parser 120 can build a complete set of hierarchical data showing the identity and relative position of all design cells in a particular design.

Detailed Description Text (26):

Parser 120 also responds to requests from first comparator module 90 to generate the hierarchical data representative of the arrangement of design cells within design database 20 that can be sent to first comparison module 90 to identify the differences between the design hierarchical data and schedule hierarchical data associated with schedule database 30 after a change has been made to schedule database 30. The generated hierarchical data will be filtered and translated by

filter 130 and translator 140 as described below before being sent to first comparator module.

Detailed Description Text (27):

Filter 130 filters, once parser 120 has generated the hierarchical data, the hierarchical data in order to generate a resultant hierarchical data that requires synchronization with schedule hierarchical data of schedule database 30. Such filters, as discussed in FIG. 2, may include commands filtering out hierarchical data related to particular views within design database 20, certain elements within a particular module, or particular levels of design cells not be synchronized with schedule database 30. For example, design database 20 may include a module that contains sub-modules which are to be designed during the course of the design project and other sub-modules that were previously designed or that are standard templates used to represent standard electrical components. In one particular filtering scheme, only those modules that are to be designed in this particular design project are to be synchronized in the databases. It may be possible that changes to design data in design database 20 do not change the design hierarchical data, it is also possible that changes may occur only to those portions of the hierarchical data that are filtered out by filter 130. Thus, design interface may send hierarchical data to first comparator module 90 that is unchanged relative to what the hierarchical data would have been before a data change to design database 20.

Detailed Description Text (33):

Schedule interface 70 includes a monitor 210 in communication with schedule database 30 and a parser 220. A filter 230 is also in communication with parser 220 and additionally with a translator 140. Translator 240 is in communication with first comparison module 90, second comparison module 100, and a command generator 250. Command generator 250 is also in communication with schedule database 30. Elements of schedule interface 70 such as monitor 110, parser 120, filter 130, translator 140, and command generator 150 may all be routines or software processes performed by a single processing device.

Detailed Description Text (35):

Additionally, after receiving update data from first comparison module 90 in response to a change in design database 20, schedule interface 70, after changing schedule database 30, will then respond as if a new change has been made to schedule database 30 that is independent of changes to design database 20. As a result, schedule interface 70 will parse, filter and translate hierarchical data and send it to second comparison module 100 so that ticket database 40 can also be updated based on the original change to design database 20.

Detailed Description Text (37):

Ticket interface 70 includes a parser 330 in communication with ticket database 30 and a filter 340 in communication with parser 330 and with a translator 310. Translator 310 is in communication with second comparison module 100 and a command generator 320. Command generator 320 is also in communication with ticket database 30. Elements of ticket interface 70 such as parser 330, filter 340, translator 310, and command generator 320 may all be routines or software processes performed by a single processing device.

Detailed Description Text (38):

As discussed earlier, changes within ticket database 40 are generally highly specific and individual in nature and generally do not require corresponding changes in design database 20 or schedule database 30. However, ticket interface 80 may still respond to changes to design database 20 and schedule database 30. Thus, ticket interface 80 awaits an indication by second comparison module 100 that a change has been detected in schedule database 30. Once such a change has been so indicated, parser 330 within ticket interface 80 parses ticket database 40 to generate hierarchical data associated with ticket database 40. Filter 340 applies

filters similar to those employed in design interface 60 such that the resulting filtered hierarchical data only includes those ticket or action items related to tasks and sub-tasks that are to correspond to schedule database 30. Translator 310 then translates the resulting filtered hierarchical data into a comparison format that can be easily compared by second comparison module 100. As a result of such comparison, ticket interface 80 may receive update data from second comparison module 100. Translator 310 then translates such update data into a format specific to ticket interface. Command generator 320 generates a command, macro or other instruction to change ticket database 40 or to indicate that a change to ticket database 40 should be made by a user. As with other interfaces, ticket interface 80 may create a template of ticket or action items for ticket database 40 that may have defaults assigned or that may respond to a naming convention of tasks and sub-task within schedule database 30 in order to customize a template of action items depending on the type of task involved.

Detailed Description Text (49):

FIG. 7 is a flowchart of a method for automatically changing schedule database 30 in response to changes in design database 20. In step 400, integration module 50 monitors for changes to design data in design database 20. In step 410, integration module 50 parses design database 20 to generate design hierarchical data, representative of the position in which design data is arranged in the hierarchy of design database, in response to integration module 50 detecting a change to the design data. Then, in step 420, integration module 50 filters the design hierarchical data based on a filtering scheme imposed by integration module 50. In step 430, integration module 50 translates the filtered design hierarchical data into a predetermined format for comparison by integration module 50. In step 450, integration module 50 parses scheduling data in schedule database 30 to extract schedule hierarchical data. In step 460, integration module 50 filters the schedule hierarchical data based on the filtering scheme imposed by integration module 50. Next in step 470, integration module 50 translates the filtered schedule hierarchical data into a format for comparison by integration module 50. In step 480, integration module 50 compares the schedule hierarchical data to the design hierarchical data to determine at least one difference between the two sets of hierarchical data associated with design database 20 and schedule database 30. Then, in step 490, integration module 50 generates update data that is indicative of the difference or differences determined in step 480. In step 500, integration module 50 translates this update data into an instruction capable of automatically updating the scheduling data in schedule database 30 so that the schedule hierarchical data associated with schedule database 30 is synchronized with the hierarchical data associated with design database 20. Thus, in step 510, schedule database 30 responds to the instruction of step 500 to modify schedule database 30 in order to change scheduling data and/or to generate notification to a user of schedule database 30 that changes need to be implemented to reflect changes that have occurred in design database 20.

Detailed Description Text (50):

FIG. 8 is a flowchart of a method for automatically changing design database 30 in response to changes in schedule database 20. In step 600, integration module 50 monitors for changes to schedule data in schedule database 20. In step 610, integration module 50 parses schedule database 20 to generate schedule hierarchical data, representative of the position in which schedule data is arranged in the hierarchy of schedule database, in response to integration module 50 detecting a change to the schedule data. Then, in step 620, integration module 50 filters the schedule representative of the hierarchical data based on a filtering scheme imposed by integration module 50. In step 630, integration module 50 translates the filtered schedule hierarchical data into a format for comparison by integration module 50. In step 650, integration module 50 parses design data in design database 30 to extract design hierarchical data. In step 660, integration module 50 filters the design hierarchical data based on the filtering scheme imposed by integration module 50. Next, in step 670, integration module 50 translates the filtered design

hierarchical data into the predetermined format for comparison by integration module 50. In step 680, integration module 50 compares the design hierarchical data to the schedule hierarchical data to determine at least one difference between the two sets of hierarchical data. Then, in step 690, integration module 50 generates update data that is indicative of the difference or differences determined in step 680. In step 700, integration module 50 translates this update data into an instruction capable of automatically updating the design data in design database 30 such that hierarchical data associated with design database 30 is synchronized with hierarchical data associated with schedule database 20. Thus, in step 710, design database 30 responds to the instruction of step 700 to modify design database 30 in order to change design data and/or to generate notification to a user of design database 30 that changes need to be implemented to reflect changes that have occurred in schedule database 20.

CLAIMS:

3. The system of claim 1, wherein the integration module includes a monitor in communication with one of the design and schedule databases, the monitor operable to detect changes to the data therein.

4. The system of claim 1, wherein the integration module includes a parser operable to parse data of one of the design and schedule databases to generate hierarchical data associated therewith.

10. A system for automatically updating data of a first database in response to changes made to a second database, the system comprising: a first interface in communication with the first database; a second interface in communication with the second database, the second interface operable to detect changes to data in the second database; and a comparison module in communication with the first and second interface, the comparison module operable to compare first hierarchical data associated with the first database to second hierarchical data associated with the second database in response to a change detected by the second interface, the comparison module operable to send update data to the first interface in response to comparing the first and second hierarchical data, the first interface operable to change the data in the first database in response to the update data, the second interface includes a filter operable to filter the second hierarchical data in response to a filtering scheme.

13. The apparatus of claim 10, wherein the second interface includes a parser operable to generate the second hierarchical data associated with the second database.

19. A method of project management integration, the method comprising: detecting a change to data of a first database; parsing data of the first database to generate the first hierarchical data associated with the first database; filtering the first hierarchical data in response to a filtering scheme; translating the filtered first hierarchical data into a comparison format; obtaining second hierarchical data associated with the second database in response to the detected change; comparing the first hierarchical data to the second hierarchical data and identifying at least one difference between the first and second of hierarchical data; and automatically generating and sending an update command to the second database in response to the at least one identified difference.

20. The method of claim 19, wherein detecting the change comprises polling the first database.

21. The method of claim 19, wherein detecting the change comprises detecting an interrupt generated by the first database.

[Previous Doc](#)

[Next Doc](#)

[Go to Doc#](#)